BUNDESREPUBLIK DEUTSCHLAND

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen: 102 36 796.5

Anmeldetag: 8. August 2002

Anmelder/Inhaber: Christian Scheideler, Paderborn/DE; André Brinkmann, Hövelhof/DE; Dr. Friedhelm Meyer auf der Heide, Delbrück/DE; Dr. Ulrich Rückert, Soest/DE.

Bezeichnung: Verfahren und Anordnung zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet sowie ein entsprechendes Computerprogramm-Erzeugnis und ein entsprechendes computerlesbares Speichermedium

IPC: G 06 F 15/173

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 10. Oktober 2003
Deutsches Patent- und Markenamt
Der Präsident
Im Auftrag
Faust

A 9161
03/00
EDV-L

Verfahren und Anordnung zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet sowie ein entsprechendes Computerprogramm-Erzeugnis und ein entsprechendes computerlesbares Speichermedium

Beschreibung

Die vorliegende Erfindung betrifft ein Verfahren und eine Anordnung zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet sowie ein entsprechendes Computerprogramm-Erzeugnis und ein entsprechendes computerlesbares Speichermedium, welche insbesondere einsetzbar sind für die Verteilung und das Wiederauffinden von Daten in fehlertoleranten sowie fehlerbehafteten Systemen, wie beispielsweise Speichernetzwerke, einem Intranet oder dem Internet.

Die Organisation von mehreren Datenspeichersystemen als effizientes und flexibles Speichersystem erfordert die Lösung zahlreicher Aufgaben. Eine der wichtigsten ist es, eine geeignete Datenplatzierung, d. h. eine geeig-

nete Strategie zur Verteilung der Datenblöcke über das Speichersystem zu finden, die einen schnellen Zugriff auf die Daten und eine hohe Sicherheit gegen Datenverlust erlaubt. Im Rahmen der Beschreibung wird im Folgenden zwischen einer Menge von auf die Datenblöcke zugreifenden Einheiten, den Clients, und einer Menge von Einheiten, die Datenblöcke ausliefert, den Servern, unterschieden. Die Begriffe Server und Datenspeichersystem werden dabei synonym verwendet.

Die im Folgenden betrachteten Verfahren und Systeme dienen zum Aufbau von verteilten Datenservern und Speichernetzwerken, sowie zum Aufbau von Web-Systemen zum Caching von Daten. Ein verteilter Datenserver, bzw. ein Speichernetzwerk, besteht im Allgemeinen aus einer Menge von Computersystemen, die über ein Netzwerk mit einer Menge von Datenspeichersystemen, verbunden sind. Das Verbindungsnetzwerk zwischen den Computersystemen und den Datenspeichersystemen besteht aus einer Menge von Switches bzw. Routern, die eine Zustellung der Datenpakete zwischen kommunizierenden Einheiten sicherstellen (siehe Figur 1). Weiterhin kann das System über eine Menge von SAN-Appliances (SAN = Storage Area Network) verfügen, die an das Netzwerk angekoppelt sein können und ein Koordination zwischen den einzelnen Computersystemen und den Datenspeichersystemen sicherstellen (siehe Figur 2). Weiterhin können so genannte In-Band-Appliances zwischen die Computersysteme und die Datenspeichersysteme geschaltet werden (siehe Figur 3). In-Band-Appliances finden bei der so genannten In-Band-Virtualisierung Verwendung. Bei der In-Band-Virtualisierung befindet sich die Kontrollinstanz, die In-Band-Appliance, im Datenstrom zwischen Server und

Speicher. Die Steuerdaten wie auch die Nutzdaten laufen durch die Appliance, die den Servern als das Speichersystem selbst erscheint. Die Zuordnung von Speichersegmenten, auch als logische Volumes bezeichnet, zu jedem einzelnen Server geschieht hier. Ebenso passiert die Steuerung des Datenzugriffs über diese Appliance. Demgegenüber gibt es auch den Ansatz, die Virtualisierung über die so genannte Out-of-Band-Virtualisierung zu realisieren. In diesem Falle befindet sich die Appliance außerhalb des Datenpfades und kommuniziert über das Netzwerk (beispielsweise ein LAN) mit dem Host-Bus-Adapter (HBA) im Server, der einen speziellen Agenten benötigt. Die Appliance definiert die logischen Volumes, die ein Server benutzen darf. Die exakten Informationen über die zugehörigen logischen und physischen Blöcke speichert der Server anschließend auf seinem HBA. In-Band verfügt über den Vorteil, sich unkompliziert ins Speichernetz integrieren und warten zu lassen. Da In-Band im Datenpfad operiert, lässt sich die Datensicherheit durch eine Storage-Firewall in der SAN-Appliance mit geringem Aufwand erhöhen. Out-Band gestaltet sich auf Grund der Wechselwirkungen zwischen den zusätzlichen Agenten auf den Applikationsservern und der SAN-Appliance komplexer. Im Gegensatz zu In-Band belegt diese Methode im Switch nur wenige Ports, so dass vor allem bei großen redundant ausgelegten SANs eine höhere Skalierbarkeit zur Verfügung steht. Zudem behindert ein Ausfall der SAN-Appliance den Datenzugriff nicht. Im Falle des Einsatzes von In-Band-Appliances werden alle Lese/ Schreib-Operationen der an die In-Band-Appliances angeschlossenen Computersysteme erst von einer der In-Band-Appliances entgegengenommen, bevor sie an die Speichersysteme weitergeleitet werden. Die

Funktionalität zum Management und zur Verteilung der Daten kann dabei sowohl in die Computersysteme, in die Router, als auch in die In-Band-Appliances integriert werden. Es wird im weiteren Verlauf davon ausgegangen, dass die an ein Speichernetzwerk bzw. einen verteilten Dateiserver angeschlossenen Computersysteme über alle für das Auffinden von Daten notwendigen Informationen verfügen.

Ein Web-Cache ist eine Einheit in einem Netzwerk, die stellvertretend für einen oder mehrere Web-Server Zugriffe von Web-Clients beantwortet. Um diese Funktionalität zur Verfügung zu stellen, verfügt der Web-Cache über ein Speichersystem, auf dem Teile der Inhalte der Web-Server gespeichert werden. Speichert der Web-Cache die von einem Client angefragten Information nicht, so wird die Anfrage an einen übergeordneten Web-Cache, bzw. den ursprünglichen Web-Server weitergeleitet und von diesem beantwortet. Web-Caches erfreuen sich aus verschiedenen Gründen einer weiten Verbreitung im Internet. Durch den Einsatz eines Web-Caches kann die Latenzzeit, die zwischen dem Stellen einer Anfrage von dem Web-Client bis zu der erfolgreichen Auslieferung der Informationen an den Web-Client vergeht, signifikant reduziert werden. Dieses trifft besonders dann zu, wenn die Bandbreite zwischen dem Web-Cache und dem Web-Client größer als die Bandbreite zwischen dem Web-Server und dem Web-Client ist oder wenn die Belastung des Web-Servers so hoch ist, dass es bei der Auslieferung der Daten in dem Web-Server selbst zu Stauungen kommt. Weiterhin kann durch den Einsatz von Web-Caches der Datenverkehr im Internet reduziert werden, wodurch eine Steigerung der Lei-

stungsfähigkeit des gesamten Systems Internet erzielt werden kann.

Durch die Kooperation mehrerer Web-Caches, die an verschiedenen Orten des Internets platziert werden, kann die Leistungsfähigkeit des Internets deutlich erhöht werden. Beispielsweise für die kooperative Zusammenarbeit mehrerer Web-Caches sind das NLANR (National Laboratory of Applied Network Research) Caching-System, dass aus einer Menge von Backbone-Caches in den USA besteht, oder das Akamai Caching-System, das Caching-Services für Unternehmen auf der ganzen Welt bereitstellt.

Der Hauptunterschied in der Bereitstellung von Verfahren zum Wiederauffinden von Daten in Speichernetzwerken bzw. verteilten Dateiservern und für Web-Caches besteht darin, dass im Falle von Speichernetzwerken die angeschlossenen Computersysteme über alle Informationen bezüglich der Platzierungsstrategie verfügen, die zum Wiederauffinden der von ihnen verwendeten Daten notwendig sind. Dieses umfasst unter anderem die Anzahl und die Eigenschaften der angeschlossenen Server, respektive der Datenspeichersysteme. Im Falle von Web-Caches verfügt der Client hingegen nur über eine beschränkte Sicht des Gesamtsystems, d. h. er kennt nicht alle an das System angeschlossene Web-Caches. Werden nicht alle Daten auf allen Web-Caches gespeichert, kann dieses dazu führen, dass der Web-Client ein Datum nicht von einem Web-Cache, sondern nur direkt vom Web-Server anfordern kann, da er entweder keinen Web-Cache kennt, der die von ihm angefragten Informationen speichert, oder weil er zwar den für ihn re-

levanten Web-Cache kennt, jedoch diesen Web-Cache nicht als für dieses Datum zuständig identifizieren kann.

Um eine hohe Effizient, Skalierbarkeit und Robustheit eines Datenspeichersystems, bzw. eines Web-Caches sicherzustellen, sind eine Reihe von Anforderungen zu erfüllen. Eine geeignete Datenverwaltungsstrategie sollte:

1. jede anteilsmäßige Aufteilung der Datenblöcke auf die Speichersysteme erfüllen können. Für identische Systeme wird in der Regel die gleichmäßige Verteilung der Datenblöcke über die Systeme gefordert.
2. es ermöglichen, die Datenfragen gemäß der anteilsmäßigen Zuordnung der Datenblöcke an die Datenspeichersysteme verteilen zu können. Für den Fall unterschiedlicher Zugriffshäufigkeiten auf Datenblöcke ist dieser Punkt nicht automatisch durch Punkt 1 sichergestellt.
3. fehlertolerant sein, d. h. Ausfälle von Datenspeichersystemen ohne Datenverlust überstehen können. Die verlorenen Teile sollten in möglichst kurzer Zeit neu generiert werden können.
4. sicherstellen, dass bei einer Hinzufügung oder Wegnahme von Datenspeichersystemen nur möglichst wenige Datenblöcke replatziert werden müssen, um die oberen Punkte wieder herzustellen. Dieses sollte möglichst ohne spürbare Beeinträchtigung des laufenden Betriebs geschehen.
5. eine kompakte Speicherung und effiziente Berechenbarkeit der Platzierung sicherstellen.

Verfügt der Client nur über unvollständige Informationen über die Verteilung der Daten über die Datenspei-

chersysteme, wie z. B. der Client von Web-Caches, so muss zusätzlich der folgende Punkt unterstützt werden:

6. auch wenn der Client nur über unvollständige, bzw. falsche Informationen über den Aufbau des Speichersystems verfügt, muss die Datenplatzierungsstrategie sicherstellen, dass eine höchst mögliche Anzahl von Zugriffen auf das Speichersystem erfolgreich ist, d. h. an einen die Informationen speichernden Server gestellt werden.

Es gibt im Wesentlichen zwei Standardstrategien für die Speicherung von Daten in Festplattensystem:

1. die Verwendung einer Zeigerstruktur, die ähnlich der Verbindungsstruktur in Dateisystemen für klassische Speichermedien (wie z. B. Festplatten und Disketten) arbeitet, oder
2. die Verwendung eines virtuellen Adressraums, der ähnlich eines virtuellen Adressraums in Rechnern verwaltet wird.

Wir werden uns im Folgenden auf den zweiten Punkt beschränken und annehmen, die Daten eines Festplattensystems werden in Form eines virtuellen Adressraums gleichgroßer Datenblöcke verwaltet. Das Problem besteht also darin, eine geeignete Abbildung des virtuellen Adressraums auf die Festplatten zu finden.

Die einfachste Art der Abbildung ist das so genannte *Disk-Striping* [CPK95], das in vielen Ansätzen in unterschiedlicher Granularität verwendet wird [PGK88, TPBG93, BBBM94, BHMM93, HG92, BGMJ94, BGM95]. Diese Methode hat eine weite Verbreitung in

Festplattenfeldern (auch als RAID-Arrays [RAID = Redundant Array of Independent Disks] bezeichnet) erfahren, da viele der optionalen Platzierungsmethoden (genannt: RAID-Level) auf Disk-Striping aufbauen. Beim Disk-Striping werden die Datenblöcke des virtuellen Adressraums (oder Teilblöcke dieser Datenblöcke) zyklisch um die Festplatten gewickelt. Diese Strategie hat den Nachteil, dass sie sehr unflexibel bezüglich einer sich ändernden Anzahl an Festplatten ist. Eine Veränderung um lediglich eine Festplatte kann eine fast vollständige Neuverteilung der Datenblöcke erfordern. Aus diesem Grund sind heutige Festplattenfelder nur schlecht skalierbar. Üblicherweise werden daher Festplattensysteme mit sehr vielen Festplatten in mehrere RAID-Arrays untergliedert.

Die Verwendung von zufälligen Datenplatzierungen (mittels pseudo-zufälliger Funktionen) ist bereits von vielen Forschern als vielversprechende Alternativmethode angesehen worden [AT97, B97, SMB98, K97]. In dieser Technik werden den Datenblöcken zufällig ausgewählte Festplatten zugewiesen. Zu den ersten, die zufällige Datenplatzierungsstrategien untersucht haben, zählen Mehlhorn und Vishkin [MV83]. Insbesondere haben sie untersucht, inwiefern mehrere zufällig platzierte Kopien pro Datenblock helfen können, um Anfragen gleichmäßig auf die Speichereinheiten zu verteilen. Weitere wichtige Resultate in dieser Richtung sind z. B. von Upfal and Wigderson [UW87] und Karp, Luby und Meyer auf der Heide [KLM92] erzielt worden.

Birk [B97] hat ähnliche Datenabbildungs- und -zugriffsstrategien vorgeschlagen, aber er verwendet eine Paritätskodierung der Datenblöcke.

Weitere Arbeiten sind unter anderem von Santos und Muntz im Rahmen des RIO Datenserver-Projekts (RIO = Remote I/O) durchgeführt worden [SMB98, SM98]. Sie vergleichen die zufällige Platzierung mit traditionellen Striping-Methoden und zeigen, dass selbst in Situationen, für die Disk-Striping entwickelt worden ist (reguläre Zugriffsmuster), die zufällige Platzierung gleichwertig oder besser ist [SM98b]. Ihre zufällige Platzierung basiert auf einem zufälligen Muster fester Größe. Falls die Anzahl der Datenblöcke diese Größe übersteigt, dann wenden sie das Muster wiederholt an, um den gesamten Datenraum auf die Festplatten abzubilden. Das kann natürlich zu unangenehmen Korrelationen zwischen den Datenblöcken führen und eine Abweichung von der Gleichverteilung der Datenblöcke und Anfragen verursachen.

Bisher gibt es jedoch nur wenige Ansätze, die in der Lage sind, die Anforderungen an eine effiziente, pseudo-randomisierte Datenplatzierung zu erfüllen. Besonders Schwierigkeiten ergeben sich dann, wenn heterogene, das heißt verschieden große Datenspeichersysteme verwendet werden oder wenn Datenspeichersysteme dynamisch in ein System eingefügt oder aus dem System herausgenommen werden.

Ein erster Ansatz, um Datenblöcke dynamisch und randomisiert über Datenspeichersysteme zu verteilen, ist in [KLL+97] vorgestellt worden. Dort werden (pseudo-)zufällige Funktionen verwendet, um den Datenblöcken und Datenspeichersystemen zufällige reelle Punkte im Intervall [0,1] zuzuweisen. Ein Datenblock wird immer von dem Datenspeichersystem gespeichert, dessen Punkt

am nächsten am Punkt des Datenblocks im [0,1]-Intervall liegt. Der Vorteil dieser Strategie liegt darin, dass sie einfach zu verwalten ist und sie nur die Replatzierung einer erwartungsgemäß minimalen Anzahl an Blöcken bei einer wechselnden Anzahl an Datenspeichersystemen erfordert. Sie hat allerdings den Nachteil, dass relativ hohe Schwankungen um den Erwartungswert für die Anzahl der auf einem Datenspeichersystem zu speichernden Blöcke und der zu replatzierenden Blöcke auftreten können und dass sie nur für homogene Datenspeichersysteme effizient anwendbar ist.

In [BBS99] wurde ein Verfahren vorgestellt, das auch auf (pseudo-)zufälligen Funktionen aufbaut. Die Datenblöcke werden wie auch in [KLL+97] mittels einer solchen Funktion auf zufällige Punkte im [0,1]-Intervall abgebildet. Aber die Zuordnung des [0,1]-Intervalls auf die Datenspeichersysteme geschieht mittels einer fest vorgegebenen Abbildung, die Assimilierungsfunktion genannt wird. Diese Funktion sorgt dafür, dass jede Festplatte den gleichen Anteil des [0,1]-Intervalls zugewiesen bekommt. Damit kann gewährleistet werden, dass nicht nur die benutzten Datenblöcke des virtuellen Adressraums sondern auch Anfragen an diese Blöcke gleichmäßig über die Festplatten verteilt werden können. Ein Vorteil dieses Verfahrens im Vergleich zu [KLL+98] liegt darin, dass die Assimilierungsfunktion die Daten mit wesentlich geringeren Abweichungen von der Gleichverteilung über die Datenspeichersysteme verteilen kann. Wie die Strategie in [KLL+98] benötigt diese Strategie nur die Replatzierung einer erwartungsgemäß minimalen Anzahl an Blöcken bei einer wechselnden Anzahl an Datenspeichersystemen. Allerdings funktio-

niert sie wie die Strategie in [KLL+98] nur gut für homogene Systeme.

Da es häufig aus Kostengründen nicht effizient ist, dass ein Speichersystem rein aus identischen Datenspeichersystemen besteht, wurden in [BSS00] auch Strategien für nichtuniforme Datenspeichersysteme entworfen. Diese basieren auf der in [BBS99] vorgestellten Strategie für identische Datenspeichersysteme. Zunächst wird angenommen, alle Systeme haben die gleiche Speicherkapazität. Auf alle die Intervallteile, die über die Kapazität einer Datenspeichersysteme hinausgehen, wird dann in einer zweiten Runde noch einmal die Strategie für identische Festplatten angewandt, allerdings diesmal nur auf die Datenspeichersysteme, die nach der ersten Platzierungsrunde noch freie Kapazitäten besitzen. Die dabei nicht unterzubringenden Intervallteile werden in einer weiteren Runde noch einmal platziert, usw., bis das komplette [0,1]-Intervall untergebracht ist. Der Hauptnachteil dieses Verfahrens besteht darin, dass es Situationen gibt, in denen deutlich mehr an Daten umplatziert werden, als minimal notwendig.

Die Aufgabe, die durch die Erfindung gelöst werden soll, besteht darin, ein Verfahren und eine Anordnung zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet sowie ein entsprechendes Computerprogramm-Erzeugnis und ein entsprechendes computerlesbares Speichermedium bereitzustellen, durch welche die vorstehend genannten Nachteile behoben werden und insbesondere eine effektive Behandlung von Speichernetzwerken, die heterogene Speichermedien umfassen, sowie eine dynamische

Skalierung von Speichernetzwerken durch Einfügen oder Herausnehmen von Speichermedien gewährleistet wird.

Diese Aufgabe wird erfindungsgemäß gelöst durch die Merkmale im kennzeichnenden Teil der Ansprüche 1, 15, 23 und 24 im Zusammenwirken mit den Merkmalen im Oberbegriff. Zweckmäßige Ausgestaltungen der Erfindung sind in den Unteransprüchen enthalten.

Ein besonderer Vorteil der Erfindung liegt darin, dass durch das Verfahren zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet die Behandlung von Änderungen im Speichernetzwerk ganz erheblich vereinfacht wird, indem eine Menge von Datenblöcken D_i (i=1, ..., m) einer Menge von Datenspeichersystemen S_j (j=1, ..., n) gemäß den folgenden Schritten zugeordnet und dort gespeichert wird:

a) der Gesamtmenge der Datenspeichersysteme wird ein virtueller Speicherraum und jedem einzelnen Datenspeichersystem S_j (j=1, ..., n) durch einen ersten Zufallsprozeß mindestens ein Teilraum I_j des virtuellen Speicherraums zugeordnet, wobei das Verhältnis zwischen dem Teilraum I_j und dem gesamten virtuellen Speicherraum wenigstens näherungsweise dem Verhältnis der auf das Datenspeichersystem S_j bzw. auf die Gesamtmenge der Datenspeichersysteme bezogenen Werte eines vorgebbaren Parameters entspricht,

b) jedem Datenblock D_i (i=1, ..., m) wird durch einen zweiten Zufallsprozeß ein (zufälliges) Element h(i) des virtuellen Speicherraums zugeordnet,

c) für jeden Datenblock D_i (i=1, ..., m) wird mindestens ein Teilraum I_k ermittelt, in dem h(i) ent-

halten ist, und der Datenblock D_i mindestens einem der durch diese(n) Teilräume (Teilraum) I_k repräsentierten Datenspeichersystem S_k zugeordnet und dort gespeichert.

Eine Anordnung zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet ist vorteilhafterweise so eingerichtet, daß sie mindestens einen Prozessor umfaßt, der (die) derart eingerichtet ist (sind), daß ein Verfahren zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet durchführbar ist, wobei die randomisierte Datenspeicherung die Verfahrensschritte gemäß einem der Ansprüche 1 bis 14 umfaßt.

Ein Computerprogrammprodukt zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet umfaßt ein computerlesbares Speichermedium, auf dem ein Programm gespeichert ist, das es einem Computer ermöglicht, nachdem es in den Speicher des Computers geladen worden ist, ein Verfahren zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet durchzuführen, wobei die randomisierte Datenspeicherung die Verfahrensschritte gemäß einem der Ansprüche 1 bis 14 umfaßt.

Um eine randomisierte Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet durchzuführen, wird vorteilhafterweise ein computerlesbares Speichermedium eingesetzt, auf dem ein Programm gespeichert ist, das es einem Computer ermöglicht, nachdem es in den Speicher des Computers geladen worden

ist, ein Verfahren zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet durchzuführen, wobei die randomisierte Datenspeicherung die Verfahrensschritte gemäß einem der Ansprüche 1 bis 14 umfaßt.

In einer bevorzugten Ausführungsform des erfindungsgemäßen Verfahrens ist vorgesehen, dass bei dem ersten und/oder zweiten Zufallsprozeß pseudo-zufällige Funktionen angewendet werden.

Als ein weiterer Vorteil erweist es sich, wenn Datenspeichersysteme S_j, deren Wert c_j des vorgebbaren Parameters einen ebenfalls vorgebbaren zweiten Wert δ übersteigt, in $\left\lfloor \frac{c_j}{\delta} \right\rfloor$ neue virtuelle Datenspeichersysteme $S_{j'}$ mit $c_{j'} = \delta$ und - falls $c_j - \left\lfloor \frac{c_j}{\delta} \right\rfloor * \delta \neq 0$ - in ein weiteres virtuelles Datenspeichersystem S_k mit $c_k = c_j - \left\lfloor \frac{c_j}{\delta} \right\rfloor * \delta$ zerlegt werden und diesen virtuellen Datenspeichersystemen durch den ersten Zufallsprozeß jeweils mindestens ein Teilraum $I_{j'}$ bzw. I_k des virtuellen Speicherraums zugeordnet wird, wobei $\lfloor a \rfloor$ den ganzzahligen Anteil einer Zahl $a \in \mathbb{R}$ beschreibt.

Des weiteren ist es von Vorteil, wenn der virtuelle Speicherraum durch das Intervall [0,1) und die Teilräume I_j durch mindestens ein in [0,1) enthaltenes Teilintervall repräsentiert werden und im ersten Zufallsprozeß durch die Anwendung einer ersten Hash-Funktion g(j) der linke Rand des Intervalls I_j ermittelt und die Länge des Intervalls gemäß $(g(j) + s * c_j)$ berechnet wird, mit:

c_j: Wert des auf das Datenspeichersystem S_j bezogenen Parameters und

s: Stretch-Faktor, der so gewählt ist, daß $s * c_j < 1$ erfüllt ist.

Von Vorteil ist es dabei, wenn der Stretch-Faktor s derart gewählt wird, dass das Intervall [0,1) vollständig durch die Teilintervalle I_j überdeckt wird.

Im zweiten Zufallsprozeß wird vorteilhafterweise durch die Anwendung einer zweiten Hash-Funktion h(i) jedem Datenblock D_i (i=1, ..., m) eine Zahl $h(i) \in [0,1)$ zugeordnet.

In einer bevorzugten Ausführungsform des Verfahrens zur randomisierten Datenspeicherung ist vorgesehen, dass der vorgebbare Parameter die physikalische Kapazität von Datenspeichersystemen oder die Anfragelast von Datenspeichersystemen beschreibt oder Abweichungen von der gewünschten Verteilung korrigieren.

In dem Fall, dass das einem Datenblock D_i zugeordnete Element h(i) in mehreren Teilräumen I_j enthalten ist, erweist es sich als vorteilhaft, dass eine uniforme Platzierungsstrategie angewendet wird, um den Datenblock D_i einem der durch die Teilräume I_j repräsentierten Datenspeichersystem zuzuordnen.

Darüber hinaus ist es von Vorteil, dass bei Änderungen mindestens eines der Werte $C=(c_1, \ldots, c_n)$ des vorgebbaren Parameters eine erneute Zuordnung der Datenblöcke D_i zu den Datenspeichersystemen S_j nach dem Verfahren zur randomisierten Datenspeicherung gemäß einem der Ansprüche 1 bis 9 unter Zugrundelegung der neuen Parameterwerte $C'=(c_1', \ldots, c_n')$ erfolgt.

In bestimmten Fällen kann es nützlich sein, bei nur geringen Änderungen von Werten des vorgebbaren Parameters keine Neuverteilung der Datenblöcke vorzunehmen. Dies wird erreicht, indem bei Änderungen mindestens eines der Werte $C=(c_1, \ldots, c_n)$ des vorgebbaren Parameters eine erneute Zuordnung der Datenblöcke D_i zu den

Datenspeichersystemen S_j nach dem Verfahren zur randomisierten Datenspeicherung gemäß einem der Ansprüche 1 bis 9 unter Zugrundelegung der neuen Parameterwerte $C' = (c_1', \ldots, c_n')$ nur erfolgt, wenn ein neuen Parameterwert c_i' sich von dem entsprechenden aktuellen Parameterwert c_i um einen vorgebbaren Wert μ unterscheidet.

Bei großen Änderungen des vorgebbaren Parameters wiederum werden Anpassungen des Systems vorteilhafterweise vorgenommen, indem bei Änderungen mindestens eines der Werte $C = (c_1, \ldots, c_n)$ des vorgebbaren Parameters in einen neuen Parameterwert $C' = (c_1', \ldots, c_n')$ stufenweise eine erneute Zuordnung der Datenblöcke D_i zu den Datenspeichersystemen S_j nach dem Verfahren zur randomisierten Datenspeicherung gemäß einem der Ansprüche 1 bis 9 erfolgt, wobei in jeder Stufe k Zwischen-Parameterwerte $C^k = (c^k_1, \ldots, c^k_n)$ mit $|c_i - c^k_i| \leq |c_i - c'_i|$ $(i = 1, \ldots, n)$ zugrundegelegt werden. Dieses Vorgehen hat den großen Vorteil, dass das System im Gegensatz zu einem direkten Update wesentlich schneller auf hohe Anfragebelastungen oder eine neue, vom Administrator gewählte Kapazitätsverteilung C'' reagieren kann, da in jedem C^i der Übergangsprozess von C nach C' abgebrochen werden kann.

Darüber hinaus ist es von Vorteil, dass zur Abspeicherung der Datenblöcke in einem Speichermedium mindestens eine Tabelle bereitgestellt wird, in denen die Zuordnung zwischen virtueller Adresse und physikalischer Adresse auf dem Speichermedium abgespeichert ist.

Ein weiterer Vorteil des erfindungsgemäßen Verfahrens zur randomisierten Datenspeicherung besteht darin, dass mehrere Datenblöcke zu einem Extent zusammengefasst werden, denen in der Tabelle eine gemeinsame physika-

lische Adresse auf dem Speichermedium zugeordnet wird, wobei die Datenblöcke eines Extents im logischen Adressraum miteinander verbunden sind, indem der erste Datenblock eines aus 2^{λ} Datenblöcken bestehenden Extents eine Adresse der Form x00...000 erhält, wobei die unteren λ Bits Null sind, der letzte Block dieses Extents die Adresse *x11...111 erhält*, wobei die untersten λ Bits Eins sind, und die physikalische Position eines Datenblocks durch eine Addition des Tabelleneintrags für den zugehörigen Extent mit den letzten λ Bits der logischen Adresse des Datenblocks gewonnen wird. Durch dieses Vorgehen wird die Anzahl von zu sichernden Tabelleneinträgen reduziert.

In einer bevorzugten Ausführungsform der Erfindung ist vorgesehen, dass die Anordnung mindestens einem Datenspeichersystem und/oder mindestens einem Computersystem, das (die) lesend und/oder schreibend auf die Speichermedien zugreift (zugreifen), und/oder mindestens eine zwischen das (die) Computersystem(e) und das (die) Datenspeichersystem(e) geschaltete Kontroller-Einheit zur Steuerung des Verfahrens randomisierten Datenspeicherung umfasst. Die Datenspeichersysteme umfassen dabei vorteilhafterweise Festplattenfelder und/oder als Web-Cashes ausgebildete Zwischenspeicher. Weiterhin stellt es sich als vorteilhaft heraus, wenn die Anordnung mindestens eine zwischen das (die) Computersystem(e) und das (die) Datenspeichersystem(e) geschaltete Kontroller-Einheit zur Steuerung des Verfahrens zur randomisierten Datenspeicherung umfasst. Dabei kann es sich als nützlich erweisen, dass das Verfahren zur randomisierten Datenspeicherung als Hardware-RAID-Verfahren in der Kontroller-Einheit implementiert ist.

In einer weiteren bevorzugten Ausführungsform der Erfindung ist vorgesehen, dass die Anordnung mindestens ein dediziertes, über Mittel zum Datenaustausch mit Speichermedien und Computersystemen der Anordnung verbundenes Computersystem (SAN-Appliance) zur Koordination der Datenspeicherung und/oder über Mittel zum Datenaustausch mit Speichermedien und Computersystemen der Anordnung verbundene Rechenressourcen (In-Band-Appliances) zur Verteilung der Datenblöcke umfasst. Ebenso stellt es einen Vorteil dar, dass die Anordnung heterogene Speichermedien umfasst.

Die Erfindung soll nachstehend anhand von zumindest teilweise in den Figuren dargestellten Ausführungsbeispielen näher erläutert werden.

Es zeigen:

Fig. 1 Aufbau eines Speichernetzwerkes,

Fig. 2 Veranschaulichung der Out-of-Band Virtualisierung des Datenraums,

Fig. 3 Veranschaulichung der In-Band Virtualisierung,

Fig. 4 Aufteilung der virtuellen Adresse eines Datenblocks zur Bestimmung der zugehörigen Festplatte und des zugehörigen Metablocks.

Wie aus dem Anforderungsprofil an die Datenverwaltungsstrategie ersichtlich wird, ist die Lösung der Aufgabenstellung im Allgemeinen davon abhängig, ob die an ein System angeschlossenen Clients 3 über alle für die Datenverteilung notwendigen Informationen verfügen. Im Folgenden wird das erfindungsgemäße Verfahren, welches nachfolgend als Share-Strategie bezeichnet wird,

vorgestellt, welches in der Lage ist, in beiden Fällen nahezu optimale Verteilungs- und Zugriffseigenschaften zu garantieren.

Nachfolgend werden kurz Voraussetzungen und Definitionen vorgestellt, die bei der Beschreibung des Ausführungsbeispiels benutzt werden.

Die Anzahl der in einem System zu speichernden Datenblöcke wird mit *m*, die Anzahl der maximal verwendbaren Datenspeichersysteme mit *N* bezeichnet. *N* wird dabei durch die Datenplatzierungsstrategie vorgegeben und ist nicht von der aktuellen Anzahl und Größe der Datenspeichersysteme abhängig. Die Anzahl der in dem System tatsächlich verfügbaren Datenspeichersysteme wird mit *n* bezeichnet. Für den Fall, dass die Anzahl der von den Datenspeichersystemen speicherbaren Datenblöcke kleiner als *m* ist, ist es erforderlich, dass ein weiteres Speichersystem zur Verfügung gestellt wird, in das aktuell nicht abbildbare Datenblöcke ausgelagert werden können.

Der Anteil der Datenblöcke, die von einem Datenspeichersystem *i* gespeichert werden können, wird als *relative Kapazität* $c_i \in [0,1]$ bezeichnet, wobei $\sum_i c_i = 1$. Die Größe der individuellen c_i kann dabei von verschiedenen Faktoren abhängen, so z. B. von der Speicherkapazität, wenn es sich um eine Festplatte handelt, oder von der Bandbreite der angeschlossenen Verbindungen bei einem Web-Cache. Zielsetzung einer Datenplatzierungsstrategie sollte es sein, dass auf jedem Datenspeichersystem *i* bei *m* zu platzierenden Datenblöcken $c_i * m$ Datenblöcke gespeichert werden. Bei der Beschreibung der umzusetzenden Techniken wird nicht davon ausgegangen, dass sich die Anzahl der Datenspei-

chersysteme in dem System verändert. Diese Situation kann dadurch modelliert werden, dass die relative Kapazität c_i eines Datenspeichersystems i, das sich zum Zeitpunkt t nicht in dem System befindet, zu diesem Zeitpunkt auf Null gesetzt wird.

Die Aufgabe der Datenverteilungsstrategie kann nun in zwei Aufgabenpunkte untergliedert werden. In einem ersten Schritt muss ein Datenblock mit seiner virtuellen Adresse einem Datenspeichersystem zugeordnet werden. Diese Zuordnung wird im Folgenden auch als *globale Datenverteilung* bezeichnet. In einem zweiten Schritt muss der Datenblock nicht nur einem Datenspeichersystem, sondern zusätzlich auch einer Position auf diesem Datenspeichersystem zugeordnet werden. Diese Zuordnung wird im Folgenden auch als *lokale Datenverteilung* bezeichnet. Die Erfindung beschäftigt sich mit dem Problem der globalen Datenverteilung. Im Rahmen der Beschreibung des erfindungsgemäßen Verfahrens werden kurz einfache lokale Datenverteilungsstrategien vorgestellt, die unsere neuen globalen Datenverteilungsstrategien ergänzen.

Eine Voraussetzung für den Einsatz der Share-Strategie ist es, dass sie als Subroutine eine Funktion verwenden kann, die das Problem der Datenverteilung für uniforme Datenspeichersysteme löst, d. h. für den Fall, dass $c_i = 1/n$ für alle i. Mögliche Strategien für den uniformen Fall sind in [KLL+97] und [BBS00] vorgestellt worden.

Die Share-Strategie wird nun im Detail beschrieben: In Share werden jedem Speichersystem ein oder mehrere Intervalle zugeordnet, deren Gesamtgröße der relativen Kapazität des Systems entspricht. Diese Intervalle

werden auf ein [0,1)-Intervall abgebildet, können sich aber im Gegensatz zu früheren Strategien mit anderen Intervallen überlappen. Jedem Datenblock wird nun mittels einer (pseudo-)zufälligen Funktion ein reeller Punkt im [0,1)-Intervall zugewiesen. Dieser Punkt kann eventuell zu mehreren Intervallen von Speichersystemen gehören. Falls dem so ist, wird eine uniforme Platzierungsstrategie verwendet, um den Datenblock einem dieser Speichersysteme zuzuweisen. Verändern sich nun die relativen Kapazitäten der Speichersysteme, so werden die Intervalllängen entsprechend angepasst.

Im Folgenden werden wir zunächst eine detaillierte Beschreibung der Share-Strategie geben und anschließend darlegen, warum sie anderen Strategien überlegen ist.

Die von der Share-Strategie verwendete Strategie für uniforme Datenspeichersysteme wird im Folgenden als Uniform(b,S) bezeichnet, wobei b die virtuelle Adresse des Datenblocks und S die Menge der Datenspeichersysteme beschreibt. Die Rückgabe der Funktion liefert das Datenspeichersystem, auf das der Datenblock b platziert wird.

Die Share-Strategie basiert auf zwei zusätzlichen Hash-Funktionen, die neben den möglicherweise für die uniforme Strategie verwendeten Hash-Funktionen bereitgestellt werden müssen. Die Hash-Funktion $h: \{1,\ldots,M\} \rightarrow [0,1)$ verteilt die Datenblöcke pseudo-zufällig über das Intervall [0,1). Eine weitere Hash-Funktion $g: \{1,\ldots, N\} \rightarrow [0,1)$ ordnet den beteiligten Datenspeichersystemen einen Punkt in dem Intervall [0,1) zu. Weiterhin werden die Parameter s, $\delta \in [1/N,1]$

verwendet, deren Bedeutung im weiteren Verlauf erläutert wird.

Es wird angenommen, dass n Datenspeichersysteme mit $(c_1, \ldots, c_n) \in [0,1)^n$ gegeben sind. Es wird dann die folgende Strategie verwendet: Für jedes Datenspeichersystem mit $c_i \geq \delta$ werden $\left\lfloor \frac{c_i}{\delta} \right\rfloor$ neue virtuelle Datenspeichersysteme i' mit $c_{i'} = \delta$ eingefügt. Entspricht die Summe der relativen Kapazitäten der virtuellen Datenspeichersysteme nicht der ursprünglichen Kapazität, wird ein zusätzliches virtuelles Datenspeichersystem j mit $c_j = c_i - \left\lfloor \frac{c_i}{\delta} \right\rfloor * \delta$ eingefügt. Datenspeichersysteme, deren Demand kleiner als δ sind, werden in ihrer ursprünglichen Form belassen und als einzelne, virtuelle Datenspeichersysteme angesehen. Durch die Transformation der Datenspeichersysteme werden maximal $n' \leq n + 1/\delta$ virtuelle Datenspeichersysteme erzeugt.

Jedem virtuellen Datenspeichersystem i wird nun ein Intervall I_i der Länge $s * c_i$ zugeordnet, das von $g(i)$ bis $(g(i) + s * c_i) \bmod 1$ reicht. Der [0,1)-Bereich wird also als Ring angesehen, um den die einzelnen Intervalle gewickelt werden. Die Konstante s wird als Stretch-Faktor bezeichnet. Um zu verhindern, dass ein einzelnes Intervall mehrfach um den Ring gewickelt wird, sollte $\delta \leq 1/s$ gewählt werden. Ein $\delta \geq 1/s$ ist möglich, erschwert jedoch die Umsetzung des Verfahrens.

Für jedes $x \in [0,1)$ sei $C_x = \{i: x \in I_i\}$ die Menge der Intervalle, in denen x enthalten ist. Die Anzahl der Elemente $c_x = |C_x|$ in dieser Menge wird als Contention bezeichnet. Da die Anzahl der Endpunkte der Intervalle

der virtuellen Datenspeichersysteme maximal $2n' \leq 2(n + 1/\delta)$ beträgt, wird das [0,1)-Intervall in maximal $2(n + 1/\delta)$ Rahmen $F_j \in [0,1)$ aufgeteilt, so dass für jeden Rahmen F_j die Menge C_x für jedes $x \in F_j$ identisch ist. Die Beschränkung der Anzahl der Rahmen ist wichtig, um die Größe der Datenstrukturen für die Share-Strategie zu begrenzen.

Die Berechnung des zu einem Datenblock zugehörigen Datenspeichersystems erfolgt nun durch den Aufruf: Uniform(b, $C_{h(b)}$).

Ein wichtiger Vorteil der Erfindung besteht wie erwähnt darin, daß sie die Behandlung von Änderungen im Speichernetzwerk 1 in äußerst einfacher Weise gestattet. Je nach Anforderung kann es sich dabei als sinnvoll erweisen, auf sich ändernde Umgebungen mit einer Adaption der Share-Strategie zu reagieren.

Bisher wurde erläutert, wie die Platzierung von Datenblöcken in einem statischen System vorzunehmen ist. Es wird nun angenommen, dass sich die Verteilung der relativen Kapazitäten in dem System von $C=(c_1, \ldots, c_n)$ auf $C'=(c_1', \ldots, c_n')$ verändert. Wie oben erläutert, umfasst dieses auch den Fall, dass neue Datenspeichersysteme in das System eintreten, bzw. Datenspeichersysteme das System verlassen. Es sind nun verschiedene Varianten denkbar, um einen Übergang von C nach C' vorzunehmen.

Variante 1: Direct Update

Die einfachste Methode besteht darin, direkt von C nach C' überzugehen und die entsprechenden Umplatzierungen vorzunehmen. Das hat den Nachteil, dass selbst bei

kleinsten Veränderungen wegen der Verwendung pseudo-zufälliger Funktionen eventuell Umplatzierungen von mehreren Datenblöcken vorgenommen werden müssen, und bei großen Veränderungen das System sich lange in einem Übergangszustand befindet, was die Aufrechterhaltung des oben genannten vierten Punktes der Anforderungen an Datenverwaltungsstrategien gefährden kann.

Variante 2: Lazy Update

Im Folgenden wird eine Strategie vorgestellt, die dafür sorgt, dass bei sehr geringen Kapazitätsveränderungen keine Daten umzuverteilen sind.

Sei $0 < \mu < 1$ eine feste Konstante, die als *Trägheit* der Share-Strategie bezeichnet wird. Die Share-Strategie ändert die relative Kapazität eines Datenspeichersystems i nur dann von c_i auf c_i', wenn $c_i' \geq (1 + \mu)c_i$ oder $c_i' \leq (1 - \mu)c_i$. Hierdurch kann die Summe der relativen Kapazitäten über alle Datenspeichersysteme von 1 abweichen, bleibt jedoch im Bereich von $1 \pm \mu$, so dass bei kleinem μ die Eigenschaften der Share-Strategie nicht gefährdet sind.

Variante 3: Smooth Update

Diese Variante ist sinnvoll für den Fall großer Kapazitätsänderungen. Falls C und C' große Kapazitätsabweichungen haben, werden zunächst Zwischenstufen $C_1, C_2, C_3, \ldots, C_t$ berechnet, so dass mit $C=C_0$ und $C'=C_{t+1}$ für jedes i in $\{0, \ldots, t\}$ C_i und C_{i+1} eng genug beisammen liegen, dass es dem System möglich ist, schnell von der einen zur anderen Kapazitätsverteilung und damit in einen stabilen Zustand überzugehen. Dieser Prozess hat den großen Vorteil, dass das System im Gegensatz zum Direct Update wesentlich schneller auf hohe

Anfragebelastungen oder eine neu vom Administrator gewählte Kapazitätsverteilung C'' reagieren kann, da in jedem C_i der Übergangsprozess von C nach C' abgebrochen werden kann.

Konkrete Umsetzungen der Verfahren werden in der weiteren Beschreibung erläutert.

Wahl der Kapazitäten:

Die Wahl der Kapazitäten für Share muss sich nicht notwendigerweise nach der physikalischen Kapazität eines Speichersystems richten. Da Share beliebige Kapazitätsverteilungen zulässt, können die Share-Kapazitäten auch dazu benutzt werden, um eine bessere Balancierung der Anfragelast vorzunehmen, um zum Beispiel Engpässe in den Verbindungen zu Speichersystemen oder in den Speichersystemen selbst zu beseitigen. Des Weiteren können sie benutzt werden, um Abweichungen von der gewünschten Verteilung (die wegen der Verwendung pseudo-zufälliger Hash-Funktionen nicht auszuschließen sind) auszugleichen. Die Share Strategie erlaubt also eine hohe Flexibilität in der Verteilung der Daten und eine hohe Robustheit, und erfüllt damit wichtige Anforderungen an ein Speichersystem.

Nachfolgend werden noch einige spezielle Aspekte des erfindungsgemäßen Verfahrens erläutert:

1. Abdeckung des [0,1)-Intervalls

Damit sichergestellt werden kann, dass die Share-Strategie jedem Datenpunkt ein Datenspeichersystem zuweisen kann, muss das [0,1)-Intervall vollständig durch

die Intervalle der virtuellen Datenspeichersysteme abgedeckt werden. Dieses kann bereits durch die Hash-Funktion g sichergestellt sein, indem nach der Verteilung der Intervalle der Datenspeichersysteme die Abdeckung überprüft wird und gegebenenfalls einzelne Intervalle verschoben werden. Bei einer zufälligen Platzierung der Intervalle durch eine pseudo-randomisierte Hash-Funktion h ist es jedoch ausreichend, einen Stretch-Faktor $s = k * \ln n$ mit $k \geq 3$ zu verwenden, so dass mit hoher Wahrscheinlichkeit die Intervalle der Datenspeicher Systeme das [0,1)-Intervall abdecken. Hohe Wahrscheinlichkeit bedeutet hier, dass die Wahrscheinlichkeit, dass ein Bereich nicht abgedeckt wird, kleiner als $1/n$ ist. Ergibt die Kontrolle der Verteilung der Intervalle, dass nicht jeder Punkt des [0,1)-Intervalls abgedeckt ist, so kann die Abdeckung durch eine Adaption des Stretch-Faktors erfolgen.

2. Benötigter Speicherplatz und Rechenkomplexität

Wird die in [KLL+97] vorgestellte Strategie als homogene Datenplatzierungsstrategie Uniform(b,S) verwendet, so liegt die erwartete Zeit, das zu einem Datenblock zugehörige Datenspeichersystem zu berechnen, in $O(1)$. Die Speicherkomplexität zur Berechnung der Share-Strategie liegt in $O(s * k * (n + 1/\delta))$. Nicht mitgezählt sind hier die Speicher- und Berechnungskomplexität der verwendeten Hash-Funktionen.

3. Güte der Verteilung

Werden pseudo-randomisierte Hash-Funktionen verwendet und wird ein Stretch-Faktor $s \geq 6 \ln(N/\sigma^2)$ mit $\sigma = \varepsilon/(1 + \varepsilon)$ gewählt, so bewegt sich der Anteil der

Datenblöcke, die von einem Datenspeichersystems i gespeichert werden, mit hoher Wahrscheinlichkeit in dem Bereich $S_i \in [(1 - \varepsilon)d_i, (1 - \varepsilon)d_i]$.

In den folgenden Abschnitten wird dargestellt, wie der Aufbau von Datenspeichersystemen mit Hilfe der Share-Strategie effizient durchgeführt werden kann. Es wird darauf hingewiesen, dass es sich dabei lediglich um Implementierungsbeispiele handelt. In einem ersten Schritt wird vorgestellt, wie die Funktionalität in ein allgemeines RAID-System integriert werden kann:

Integration der Share-Strategie in ein allgemeines RAID-System:

Die Share-Strategie kann verwendet werden, um in Systemen, die aus einer Menge von Speichermedien, aus mehreren Computersystemen und einer Kontroller-Einheit bestehen, Festplattenfelder aufzubauen. Dabei kann die Share-Strategie sowohl in dem angeschlossenen Computersystemen als Software-RAID Verfahren integriert werden, als auch in der Kontroller-Einheit als Hardware-RAID Verfahren. Die Share-Strategie ist dabei für die Zuordnung der Datenblöcke über die Festplatten zuständig, die Zuordnung des Datenblocks zu einer physikalischen Adresse auf der Festplatte wird von einer unter der Share-Strategie liegenden Strategie übernommen. Eine Möglichkeit für die Zuordnung der physikalischen Position besteht in der Bereitstellung von Tabellen, in denen eine Zuordnung zwischen virtueller Adresse und physikalischer Adresse auf der Festplatte abgespeichert wird.

Es ist dabei möglich, die Anzahl der zu sichernden Tabelleneinträge zu reduzieren, indem nicht jedem einzelnen Datenblock ein eigener Eintrag zugeordnet wird, sondern indem Blockmengen minimaler Größe, im Folgenden auch als Extents bezeichnet, über einen gemeinsamen Eintrag in der Tabelle verfügen. Bei einem Extent handelt es sich um eine Menge von Blöcken, die in dem logischen Adressraum miteinander verbunden sind. Ein Extent besteht aus 2^{λ} Blöcken. Der erste Block des Extents hat eine Adresse der Form *x00...000*, wobei die unteren λ Bits 7 durch die Ziffer Null repräsentiert sind. Der letzte Block des Extents hat die Adresse *x11...111*, wobei die untersten λ Bits 7 durch die Ziffer Eins repräsentiert sind. Die physikalische Position eines Datenblocks wird durch eine Addition des Tabelleneintrags für den zugehörigen Extent mit den unteren λ Bits 7 der logischen Adresse des Datenblocks gewonnen. Hat jeder Tabelleneintrag die Form *y00...000*, d. h. die unteren λ Bits 7 werden Null gesetzt, kann die Addition durch eine einfache ODER-Verknüpfung durchgeführt werden. Die oberen Bits 6 der virtuellen Adresse eines Datenblocks dienen also zur Berechnung des zugeordneten Speichermediums und der Bestimmung des Tabelleneintrages für den Extent, die unteren Bits 7 dienen als Offset innerhalb des Extents. Allen Datenblöcken, die über gemeinsame obere Bits 6 verfügen, wird ein Tabelleneintrag zugeordnet. Dieser Tabelleneintrag kann z. B. an der Stelle gespeichert werden, an der auch die Berechnung der Share-Strategie durchgeführt wird.

Integration der Share-Strategie in ein Speichernetzwerk 1:

Die Integration der globalen Datenverteilungsstrategien in ein Speichernetzwerk 1 geht von einer Struktur gemäß Figur 1 aus. Das Gesamtsystem besteht aus einer Menge von Datei- oder Datenbankservern, im Folgenden als Computersysteme bezeichnet, die über ein Speichernetzwerk 1 an Datenspeichersysteme 4 angeschlossen sind. Das Speichernetzwerk 1 umfaßt weiter eine Menge von Switches bzw. Routern 2, die die Zustellung der Datenpakete zwischen kommunizierenden Einheiten sicherstellen. Die Computersysteme sind in dem hier vorliegenden Kontext als Clients 3 zu betrachten, die von den Datenspeichersystemen 4 Blöcke lesen, oder auf den Datenspeichersystemen 4 Datenblöcke schreiben. Mit Hilfe der Share-Strategie kann jede beliebige Teilmenge *M* der an das Speichernetzwerk 1 angeschlossenen Speichersysteme 4 wie ein einziger logischer Speicherpool verwaltet werden, der über einen linearen Adressraum verfügt. Die Menge der Speichersysteme 4 kann dabei in mehrere kleinere oder einen großen Speicherpool aufgeteilt werden, wobei keine der Speichersysteme 4 mehr als einem Speicherpool zugeordnet werden sollte. Es wird im Folgenden nur der Fall betrachtet, dass das System aus einem Speicherpool besteht.

Aus einem Speicherpool können mehrere virtuelle Speichersysteme aufgebaut werden, wobei jedes dieser virtuellen Speichersysteme gemäß der Share-Strategie verwaltet wird. Besteht ein Speicherpool aus einer Teilmenge *M* der Speichersysteme, so erfolgt der Aufruf der Share-Strategie für die logischen Speichersysteme gemäß der gesamten Teilmenge *M*. Jedem virtuellen Speichersystem wird eine Speicher-Policy zugeordnet die Eigenschaften wie physikalische Blockgröße und Redundanz umfasst. Diese Zuordnung kann separat für jedes

virtuelle Speichersystem oder einmal für den gesamten Speicherpool erfolgen. Nachdem Daten auf eine virtuelle Festplatte geschrieben wurden, kann die Speicher-Policy im Allgemeinen nicht mehr verändert werden.

Wird von einem Computersystem auf einen Extent zugegriffen, der bisher von dem Computersystem noch nicht verwendet wurde und für den kein Tabelleneintrag in diesem Computersystem vorliegt, muss ein neuer Tabelleneintrag allokiert werden. Die Allokation kann auf zwei Arten erfolgen:

1. Das Computersystem fragt bei einer zentralen Instanz, die über globales Wissen über alle Tabelleneinträge verfügt, nach einem Tabelleneintrag für das Extent,
2. Auf jedem Speichersystem 4 ist ein Bereich reserviert, der eine Zuordnung zwischen virtueller Adresse und physikalischer Adresse vornimmt. Das Computersystem sucht zuerst nach der virtuellen Adresse des Extents. Falls diese Adresse noch nicht reserviert ist, sucht das Computersystem nach einer noch freien Adresse auf dem Speichersystem 4.

Wird die Koordination nicht durch eine zentrale Instanz vorgenommen, so muss diese Aufgabe nach Figur 1 von einem oder mehreren der angeschlossenen Computersysteme übernommen werden. Weiterhin können jedoch auch ein oder mehrere dedizierte Geräte, die als SAN-Appliances 5 bezeichnet werden, zur Koordination der Computersysteme gemäß Figur 2 an das Speichernetzwerk 1 angeschlossen werden. Neben der Entlastung der Computersysteme um die Koordination kann durch den Einsatz von SAN-Appliances 5 sichergestellt werden, dass alle

angeschlossenen Computersysteme die gleiche Sicht auf die Speichersysteme 4 haben, d. h. zum gleichen Zeitpunkt über das Verlassen bzw. Hinzukommen von Speichersystemen 4 informiert werden.

Die SAN-Appliance 5 bietet somit eine Reihe von Schnittstellen, über die Informationen zwischen dem SAN-Appliances 5 und den Client-Rechnern 3 ausgetauscht werden können. Diese umfassen:

- Anfrage der Grundkonfiguration von jedem Client 3,
- Anfrage nach neuen Extents von jedem Client 3,
- Information der Clients 3 über Veränderungen der Infrastruktur.

Das Share-Verfahren kann auch in so genannte In-Band-Appliances integriert werden (siehe Figur 3). Bei den In-Band-Appliances handelt es sich um dedizierte Systeme, die eine Transformation der logischen Adresse eines Datenblocks, die sie von den angeschlossenen Computersystemen erhalten, in die physikalische Adresse vornehmen. Der Einsatz von In-Band-Appliances ist dann notwendig, wenn die Funktionalität der Share-Strategie nicht in die Computersysteme direkt integriert werden kann, da keine Software-Version der Share-Strategie für diese Computersysteme verfügbar ist oder die Leistung der angeschlossenen Computersysteme nicht ausreichend groß ist, um die Transformation der logischen Adressen in die physikalischen Adressen durchzuführen.

Eine In-Band-Appliance verhält sich aus Sicht der Speichersysteme 4 wie ein angeschlossenes Computersystem, aus der Sicht der an die In-Band-Appliance

angeschlossenen Computersysteme wie ein physikalisches Speichersystem.

In dem Speichernetzwerk 1 können In-Band-Appliances mit Computersystemen, in denen die Share-Strategie ausgeführt wird, gemischt werden.

Aufbau von Internetsystemen mit Hilfe der Share-Strategie:

Die Problemstellung beim Aufbau von Systemen zur Auslieferung von Datenobjekten über das Internet unterscheidet sich von dem Aufbau von Speichersystemen in sofern, dass Clients 3 in dem Internet keine globale Sicht über alle verfügbaren Web-Server und Web-Caches in dem System haben. Soll ein Datum von einem Web-Cache gelesen werden, um die teilnehmenden Web-Server zu entlasten, muss also sichergestellt werden, dass der Client 3 mindestens einen zu einem Datenobjekt gehörenden Web-Cache kennt und das zu lesende Datenobjekt auch dem richtigen Web-Cache zuordnen kann.

Diese Aufgabenstellung kann im Allgemeinen nicht gelöst werden, ohne dass von einem Datenobjekt mehrere Kopien angelegt werden, die über die Web-Caches gemäß einer vorgegebenen Platzierungsstrategie verteilt werden. Werden von einem System von jedem Datenobjekt k Kopien gespeichert, so fragt der Client 3 nacheinander oder gleichzeitig bei den k Web-Caches nach, von denen er glaubt, dass sie eine Kopie des Datenobjektes speichern. Hält einer der Web-Caches eine Kopie des Daten-Objektes, so wird diese Kopie anschließend von dem Client 3 gelesen.

Die Anzahl der notwendigen Kopien, damit ein Client 3 einem Datenobjekt mindestens einen Web-Cache zuordnet, der auch dieses Datenobjekt speichert, ist von der verwendeten Verteilungsstrategie und den relativen Kapazitäten $C = (c_1, \ldots, c_n)$ der Web-Caches abhängig. Weiterhin ist sie von der Sicht $V = (v_1, \ldots, v_n)$ des Clients 3 abhängig, das heißt von den relativen Größen der Web-Caches, die der Client 3 zu kennen glaubt. Die Konsistenz κ_V der Sicht eines Clients 3 wird wie folgt definiert:

$$\kappa_V = \sum_{i=1}^{n} \min[v_i, c_i]$$

Es kann gezeigt werden, dass bei Verwendung der Share-Strategie die Verwendung von $\Theta(\log N)$ Kopien ausreichend ist, um mit einer Wahrscheinlichkeit von größer als $\left(1-\frac{1}{n}\right)$ garantieren zu können, das mindestens für ein Datenobjekt der Web-Cache, der von der Share-Strategie berechnet wird, für C und V derselbe ist.

Die Erfindung ist nicht beschränkt auf die hier dargestellten Ausführungsbeispiele. Vielmehr ist es möglich, durch Kombination und Modifikation der genannten Mittel und Merkmale weitere Ausführungsvarianten zu realisieren, ohne den Rahmen der Erfindung zu verlassen.

Bezugszeichenliste

1 Speichernetzwerk

2 Switches bzw. Router

3 Client

4 Datenspeichersystem

5 SAN-Appliance

6 obere Bits

7 untere Bits

Referenzen

[AT97] J. Alemany und J.S. Thathachar, "Random Striping News on Demand Server", Technischer Report der University of Washington, Department of Computer Science and Engineering, 1997

[B97] Y. Birk, "Random RAIDs with Selective Exploitation of Redundancy for High Performance Video Servers", In Proceedings of 7th International Workshop on Network and Operating System Support for Digital Ausio and Video, 1997

[BBBM94] M. Blaum, J. Brady, J. Bruck und J. Menon, EVENODD: An Optimal Scheme for Tolerating Double Disk Failures in RAID Architectures", In Proceedings of the 21st Annual International Symposium on Computer Architecture, Seiten 245-254, 1994

[BBS99] P. Berenbrink, A. Brinkmann und C. Scheideler, "Design of the PRESTO Multimedia Data Storage Network", In Proceedings of the Workshop on Communication and Data Management in Large Networks (INFORMATIK 99), 1999

[BGM95] S. Berson, L. Golubchik und R.R. Muntz; "Fault Tolerant Design of Multimedia Servers", In SIGMOD Record (ACM Special Interest Group on Management of Data), 19(2):364-375,1995 [BGMJ94] S. Berson, S. Ghandeharizadeh, R.R. Muntz und x. Ju, "Staggered Striping in Multimedia Systems", In Proceedings of the 1994 ACM Conference on Management of Data (SIGMOD), Seiten 79-90,

1994

[BHMM93] M.Blaum, H.T. Hao, R.L. Mattsoll und J.M. Menon, "Method and Means for Encoding and Rebuilding Data Contents of up to two unavailable DASDs in an in an Array of DASDs", US Patent No. 5,271,012, Dezember 1993

[BSS00] A. Brinkmann, K. Salzwedel und C. Scheideler: "Efficient, Distributed Data Placement for Storage Area Networks", In Proceedings of the 12th Symposium on Parallel Algorithms and Architectures (SPAA 2000), 2000

[CPK95] A. L. Chervenak, D. A. Pattersoll und R. H. Katz, "Choosing the best storage system video service", In Proceedings of the third ACM International Multimedia Conference and Exhibition, Seiten 109-120, 1996

[HG92] M. Holland und G. Gibson, "Parity Declustering for Continuous Operation in Redundant Disk Arrays", In Proceedings of the Fifth International Conference on Architectural Support for Programming Languages ans Operating Systems, Seiten 23-35, 1992

[K97] J. Korst, "Random Duplicated Assignment: An Alternative to Striping in Video Servers", In Proceedings of the Fifth ACM International Multimedia Conference, Seiten 219-226, 1997

[KLL+97] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin und R. Panigrahy: "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web", In Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory

of Computing (STOC), Seiten 654-663, 1997

[KLM92] R. Karp, M. Luby und F. Meyer auf der Heide, "Efficient PRAM Simulation on a Distributed Memory Machine, In Proceedings of the 24th ACM Symposium on Theory of Computing, S. 318-326, 1992

[MV83] K. Mehlhorn und U Vishkin, "Randomized and deterministic simulation of PRAMs by parallel machines with restricted granularity of parallel memories", In Proceedings of 9th Workshop on Graph Theoretic Concepts in Computer Science, 1983

[PGK88] D.A. Patterson, G. Gibson und R.H. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)", In Proceedings of the 1988 ACM Conference on Management of Data (SIGMOD), Seiten 109-116, 1988

[SM98] J.R. Santos und R.R. Muntz, "Performance Analysis of the RIO Multimedia Storage System with Heterogeneous Disk Configuration", In Proceedings of ACM Multimedia 98, Seiten 303-308, 1998

[SM98b] J.R. Santos und R.R. Muntz" "Comparing Random Data Allocation and Data Striping in Multimedia Servers", Technischer Report, University of California, Los Angeles, Computer Science Department, 1998

[SMB98] J.R. Santos, R.R. Muntz und S. Berson, " A Parallel Disk Storage System for Realtime Multimedia Applications", International Journal of Intelligent Systems, 13(12): 1137-1174, 1998

[TPBG93] F.A. Tobagi, J. Pang, R. Baird und M. Gang, "Streaming RAID: A Disk Array Management

System for Video Files", In Proceedings of Computer Graphics (Multimedia '93 Proceedings), Seiten 393-400,1993

[UW87] E. Upfal und A. Wigderson, "How to Share memory in a distributed system", Journal of the ACM, 34(1): 116-127,1987

Patentansprüche

1. Verfahren zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet,

dadurch gekennzeichnet, daß

eine Menge von Datenblöcken D_i (i=1, ..., m) einer Menge von Datenspeichersystemen S_j (j=1, ..., n) gemäß den folgenden Schritten zugeordnet und dort gespeichert wird:

a) der Gesamtmenge der Datenspeichersysteme wird ein virtueller Speicherraum und jedem einzelnen Datenspeichersystem S_j (j=1, ..., n) durch einen ersten Zufallsprozeß mindestens ein Teilraum I_j des virtuellen Speicherraums zugeordnet, wobei das Verhältnis zwischen dem Teilraum I_j und dem gesamten virtuellen Speicherraum wenigstens näherungsweise dem Verhältnis der auf das Datenspeichersystem S_j bzw. auf die Gesamtmenge der Datenspeichersysteme bezogenen Werte eines vorgebbaren Parameters entspricht,

b) jedem Datenblock D_i (i=1, ..., m) wird durch einen zweiten Zufallsprozeß ein (zufälliges) Element h(i) des virtuellen Speicherraums zugeordnet,

c) für jeden Datenblock D_i (i=1, ..., m) wird mindestens ein Teilraum I_k ermittelt, in dem h(i) enthalten ist, und der Datenblock D_i mindestens einem der durch diese(n) Teilräume (Teilraum) I_k repräsentierten Datenspeichersystem S_k zugeordnet und dort gespeichert.

2. Verfahren nach Anspruch 1,

dadurch gekennzeichnet, daß

bei dem ersten und/oder zweiten Zufallsprozeß pseudo-zufällige Funktionen angewendet werden.

3. Verfahren nach einem der Ansprüche 1 oder 2,

dadurch gekennzeichnet, daß

Datenspeichersysteme S_j, deren Wert c_j des vorgebbaren Parameters einen ebenfalls vorgebbaren zweiten Wert δ übersteigt, in $\left\lfloor \frac{c_j}{\delta} \right\rfloor$ neue virtuelle Datenspeichersysteme $S_{j'}$ mit $c_{j'} = \delta$ und - falls $c_j - \left\lfloor \frac{c_j}{\delta} \right\rfloor * \delta \neq 0$ - in ein weiteres virtuelles Datenspeichersystem S_k mit $c_k = c_j - \left\lfloor \frac{c_j}{\delta} \right\rfloor * \delta$ zerlegt werden und diesen virtuellen Datenspeichersystemen durch den ersten Zufallsprozeß jeweils mindestens ein Teilraum $I_{j'}$ bzw. I_k des virtuellen Speicherraums zugeordnet wird, wobei $\lfloor a \rfloor$ den ganzzahligen Anteil einer Zahl $a \in \mathbb{R}$ beschreibt.

4. Verfahren nach einem der vorangehenden Ansprüche,

dadurch gekennzeichnet, daß

der virtuelle Speicherraum durch das Intervall [0,1) und die Teilräume I_j durch mindestens ein in [0,1) enthaltenes Teilintervall repräsentiert werden.

5. Verfahren nach einem der vorangehenden Ansprüche,

dadurch gekennzeichnet, daß

im ersten Zufallsprozeß durch die Anwendung einer ersten Hash-Funktion g(j) der linke Rand des Intervalls I_j ermittelt und die Länge des Intervalls gemäß $(g(j) + s * c_j)$ berechnet wird, mit:

c_j: Wert des auf das Datenspeichersystem S_j bezogenen Parameters und

s: Stretch-Faktor, der so gewählt ist, daß $s * c_j < 1$ erfüllt ist.

6. Verfahren nach einem der vorangehenden Ansprüche,

dadurch gekennzeichnet, daß

der Stretch-Faktor s derart gewählt wird, dass das Intervall [0,1) vollständig durch die Teilintervalle I_j überdeckt wird.

7. Verfahren nach einem der vorangehenden Ansprüche,

dadurch gekennzeichnet, daß

im zweiten Zufallsprozeß durch die Anwendung einer zweiten Hash-Funktion h(i) jedem Datenblock D_i (i=1, ..., m) eine Zahl $h(i) \in [0,1)$ zugeordnet wird.

8. Verfahren nach einem der vorangehenden Ansprüche,

dadurch gekennzeichnet, daß

der vorgebbare Parameter

- die physikalische Kapazität von Datenspeichersystemen oder
- die Anfragelast von Datenspeichersystemen beschreibt oder
- Abweichungen von der gewünschten Verteilung korrigieren.

9. Verfahren nach einem der vorangehenden Ansprüche, dadurch gekennzeichnet, daß

in dem Fall, dass das einem Datenblock D_i zugeordnete Element $h(i)$ in mehreren Teilräumen I_j enthalten ist, eine uniforme Platzierungsstrategie angewendet wird, um den Datenblock D_i einem der durch die Teilräume I_j repräsentierten Datenspeichersystem zuzuordnen.

10. Verfahren nach einem der vorangehenden Ansprüche, dadurch gekennzeichnet, daß

bei Änderungen mindestens eines der Werte $C=(c_1, \ldots, c_n)$ des vorgebbaren Parameters eine erneute Zuordnung der Datenblöcke D_i zu den Datenspeichersystemen S_j nach dem Verfahren zur randomisierten Datenspeicherung gemäß einem der Ansprüche 1 bis 9 unter Zugrundelegung der neuen Parameterwerte $C'=(c_1', \ldots, c_n')$ erfolgt.

11. Verfahren nach einem der vorangehenden Ansprüche, dadurch gekennzeichnet, daß

bei Änderungen mindestens eines der Werte $C=(c_1, \ldots, c_n)$ des vorgebbaren Parameters eine erneute Zuordnung der Datenblöcke D_i zu den Datenspeichersystemen S_j nach dem Verfahren zur randomisierten Datenspeicherung gemäß einem der Ansprüche 1 bis 9 unter Zugrundelegung der neuen Parameterwerte $C'=(c_1', \ldots, c_n')$ nur erfolgt, wenn ein neuen Parameterwert c_i' sich von dem entsprechenden aktuellen Parameterwert c_i um eine vorgebbare Konstante μ unterscheidet.

12. Verfahren nach einem der vorangehenden Ansprüche,

dadurch gekennzeichnet, daß

bei Änderungen mindestens eines der Werte $C=(c_1, \ldots, c_n)$ des vorgebbaren Parameters in einen neuen Parameterwert $C'=(c_1', \ldots, c_n')$ stufenweise eine erneute Zuordnung der Datenblöcke D_i zu den Datenspeichersystemen S_j nach dem Verfahren zur randomisierten Datenspeicherung gemäß einem der Ansprüche 1 bis 9 erfolgt, wobei in jeder Stufe k Zwischen-Parameterwerte $C^k=(c^k_1, \ldots, c^k_n)$ mit $|c_i - c^k_i| \leq |c_i - c'_i|$ $(i = 1, \ldots, n)$ zugrundegelegt werden.

13. Verfahren nach einem der vorangehenden Ansprüche,

dadurch gekennzeichnet, daß

zur Abspeicherung der Datenblöcke in einem Speichermedium mindestens eine Tabelle bereitgestellt wird, in denen die Zuordnung zwischen virtueller Adresse und physikalischer Adresse auf dem Speichermedium abgespeichert ist.

14. Verfahren nach Anspruch 13,

dadurch gekennzeichnet, daß

mehrere Datenblöcke zu einem Extent zusammengefasst werden, denen in der Tabelle eine gemeinsame physikalische Adresse auf dem Speichermedium zugeordnet wird, wobei die Datenblöcke eines Extents im logischen Adressraum miteinander verbunden sind, indem der erste Datenblock eines aus 2^λ Datenblöcken bestehenden Extents eine Adresse der Form x00...000 erhält, wobei die unteren λ

Bits durch die Ziffer Null repräsentiert sind, der letzte Block dieses Extents die Adresse *x11...111 erhält,* wobei die untersten λ Bits durch die Ziffer Eins repräsentiert sind, und die physikalische Position eines Datenblocks durch eine Addition des Tabelleneintrags für den zugehörigen Extent mit den letzten λ Bits der logischen Adresse des Datenblocks gewonnen wird.

15. Anordnung mit mindestens einen Prozessor, der (die) derart eingerichtet ist (sind), daß ein Verfahren zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet durchführbar ist, wobei die randomisierte Datenspeicherung die Verfahrensschritte gemäß einem der Ansprüche 1 bis 14 umfasst.

16. Anordnung nach Anspruch 15,

dadurch gekennzeichnet, daß

die Anordnung

- mindestens einem Datenspeichersystem und/oder
- mindestens einem Computersystem, das (die) lesend und/oder schreibend auf die Speichermedien zugreift (zugreifen), und/oder
- mindestens eine zwischen das (die) Computersystem(e) und das (die) Datenspeichersystem(e) geschaltete Kontroller-Einheit zur Steuerung des Verfahrens randomisierten Datenspeicherung

umfasst.

17. Anordnung nach Anspruch 16,

dadurch gekennzeichnet, daß

das Datenspeichersystem
- Festplattenfelder und/oder
- als Web-Cashes ausgebildete Zwischenspeicher

umfasst.

18. Anordnung nach einem der Ansprüche 15 bis 17,

dadurch gekennzeichnet, daß

die Anordnung mindestens eine zwischen das (die) Computersystem(e) und das (die) Datenspeichersystem(e) geschaltete Kontroller-Einheit zur Steuerung des Verfahrens zur randomisierten Datenspeicherung umfasst.

19. Anordnung nach Anspruch 18,

dadurch gekennzeichnet, daß

die Anordnung mindestens ein über die Kontroller-Einheit auf die Speichermedien zugreifendes Computersystem umfasst.

20. Anordnung nach einem der Ansprüche 15 bis 19,

dadurch gekennzeichnet, daß

das Verfahren zur randomisierten Datenspeicherung als Hardware-RAID-Verfahren in der Kontroller-Einheit implementiert ist.

21. Anordnung nach einem der Ansprüche 15 bis 20,

dadurch gekennzeichnet, daß

die Anordnung
- mindestens ein dediziertes, über Mittel zum Datenaustausch mit Speichermedien und Computersystemen der Anordnung verbundenes Computer-

system (SAN-Appliance) zur Koordination der Datenspeicherung und/oder
- über Mittel zum Datenaustausch mit Speichermedien und Computersystemen der Anordnung verbundene Rechenressourcen (In-Band-Appliances) zur Verteilung der Datenblöcke

umfasst.

22. Anordnung nach einem der Ansprüche 15 bis 21, dadurch gekennzeichnet, daß die Anordnung heterogene Speichermedien umfasst.

23. Computerprogrammprodukt, das ein computerlesbares Speichermedium umfaßt, auf dem ein Programm gespeichert ist, das es einem Computer ermöglicht, nachdem es in den Speicher des Computers geladen worden ist, ein Verfahren zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet durchzuführen, wobei die randomisierte Datenspeicherung die Verfahrensschritte gemäß einem der Ansprüche 1 bis 14 umfaßt.

24. Computerlesbares Speichermedium, auf dem ein Programm gespeichert ist, das es einem Computer ermöglicht, nachdem es in den Speicher des Computers geladen worden ist, ein Verfahren zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet durchzuführen, wobei die randomisierte Datenspeicherung die Verfahrensschritte gemäß einem der Ansprüche 1 bis 14 umfaßt.

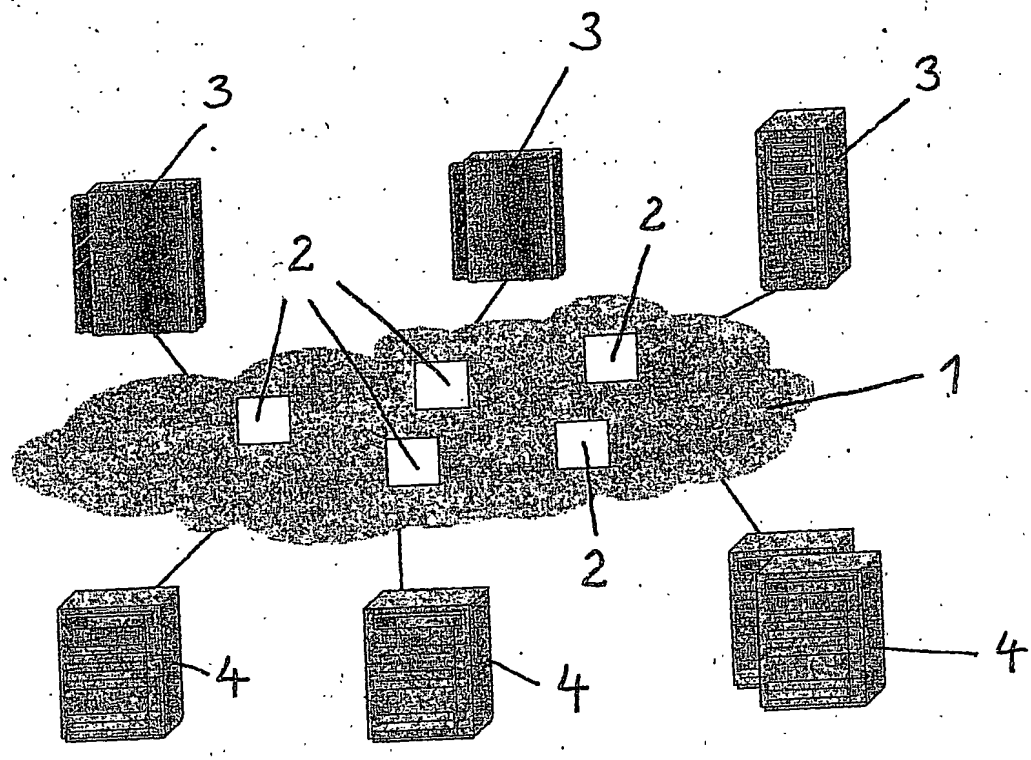

Fig. 1

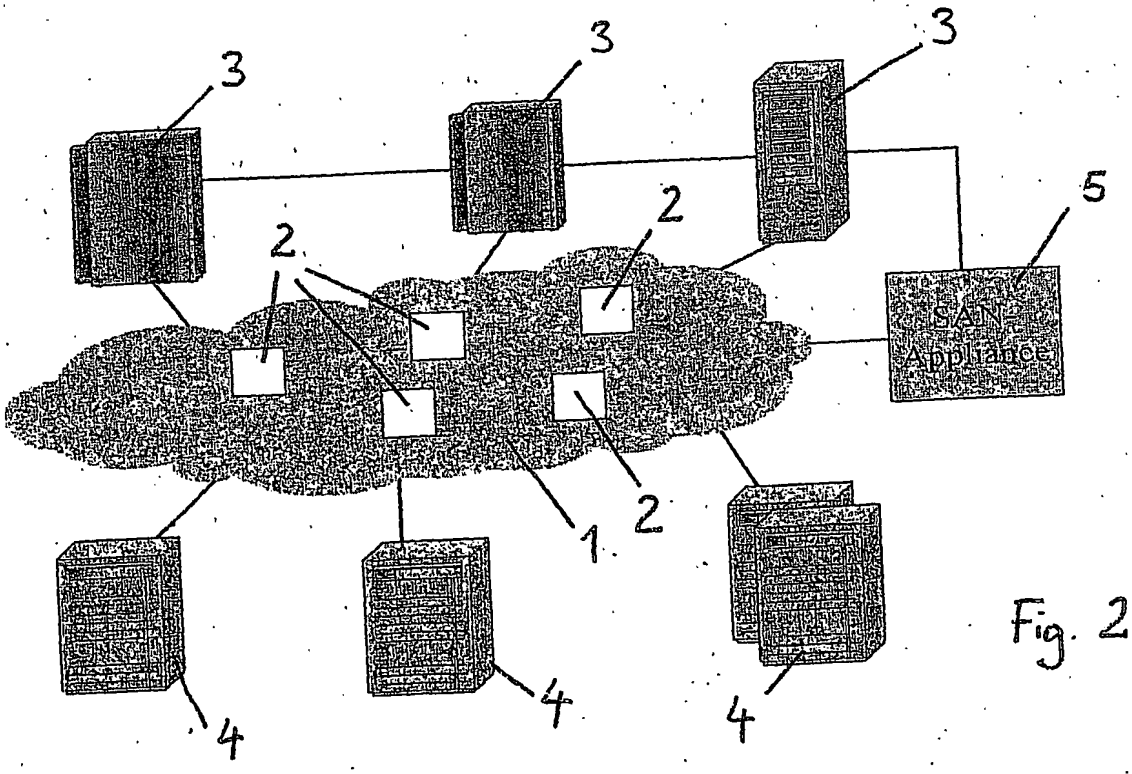

Fig. 2

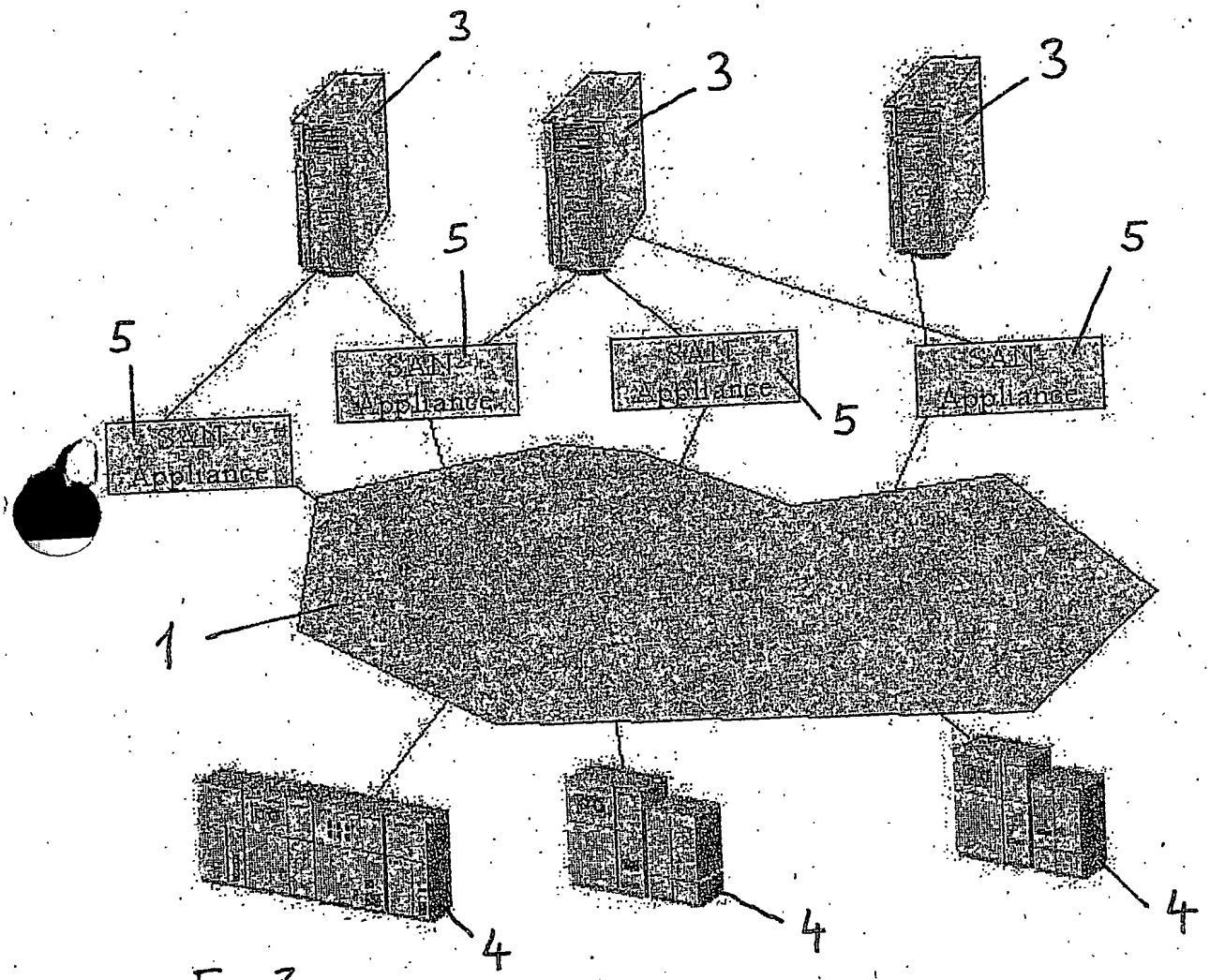

Fig. 3

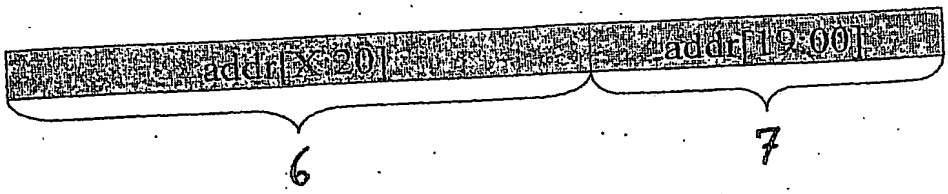

Fig. 4

Zusammenfassung

Die vorliegende Erfindung beschreibt ein Verfahren und eine Anordnung zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet sowie ein entsprechendes Computerprogramm-Erzeugnis und ein entsprechendes computerlesbares Speichermedium, welche insbesondere einsetzbar sind für die Verteilung und das Wiederauffinden von Daten in fehlertoleranten sowie fehlerbehafteten Systemen, wie beispielsweise Speichernetzwerke oder dem Internet.

Hierfür wird vorgeschlagen, dass bei dem Verfahren zur randomisierten Datenspeicherung in Speichernetzwerken und/oder einem Intranet und/oder dem Internet jedem Speichersystem ein oder mehrere Intervalle zugeordnet werden, deren Gesamtgröße der relativen Kapazität des Systems entspricht. Diese Intervalle werden auf ein [0,1)-Intervall abgebildet, können sich aber im Gegensatz zu früheren Strategien mit anderen Intervallen überlappen. Jedem Datenblock wird nun mittels einer (pseudo-)zufälligen Funktion ein reeller Punkt im [0,1)-Intervall zugewiesen. Dieser Punkt kann eventuell zu mehreren Intervallen von Speichersystemen gehören. Falls dem so ist, wird eine uniforme Platzierungsstrategie verwendet, um den Datenblock einem dieser Speichersysteme zuzuweisen. Verändern sich nun die relativen Kapazitäten der Speichersysteme, so werden die Intervalllängen entsprechend angepasst.

- Fig. 1 -

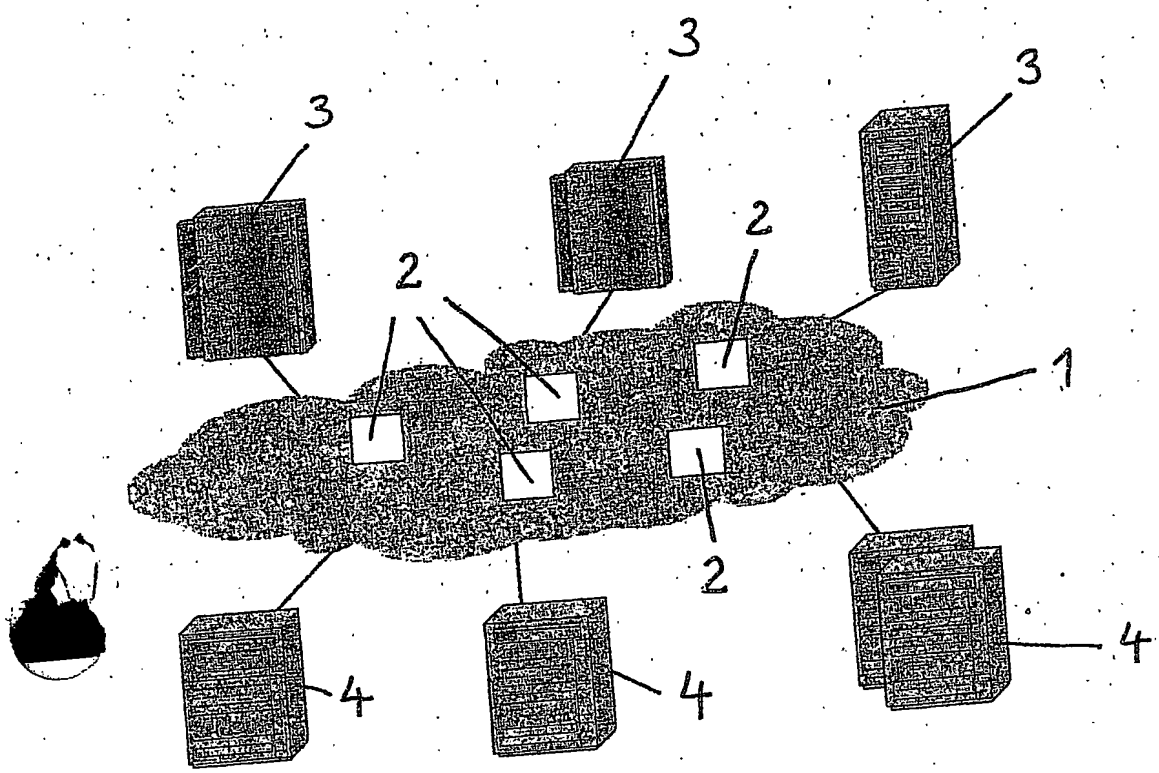

Fig. 1

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☑ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☑ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☑ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.